

Graph based KNN Variants for classifying Texts

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we propose the graph based KNN variants, and apply them to the text categorization. The initial KNN algorithm was previously modified into the version which received a graph, instead of a numerical vector, as an approach to the task. In case of numerical vector-based versions, we mention the three KNN variants: one where the selected nearest neighbors are discriminated by their similarities, one where the attributes are discriminated by their correlations with the target outputs, and one where the training examples are discriminated by their credits. We modify the three KNN variants into the graph-based versions as the text categorization tools, as well as the initial KNN version. This research is aimed to improve the classification performance by proposing the modified versions of the KNN variants.

I. INTRODUCTION

The text categorization is the task where texts are classified into one or some among the predefined categories based on their contents. A text is given as an entity in this task, it is decomposed into paragraphs, and a paragraph is decomposed into sentences. When applying a machine learning algorithm to the text classification, it is trained with the sample texts which are labeled with their own categories, and novice texts are classified by it. The task in this research is the hard text categorization where categories are predefined as a list and only one category is assigned to each text, and it will be expanded into the soft text categorization or the hierarchical text categorization in subsequent research. In future, the cooperation between the text classification and the word classification will be proposed for improving their performances by their synergy effects.

This research is motivated by the better performance of the graph-based version of KNN algorithm in applying it to the text categorization. The main problems in encoding texts into numerical vectors were solved by encoding them into graphs as alternatives to numerical vectors. The similarity metric between graphs is defined based on the similarity between edges, and the KNN algorithm was modified into the graph-based version as the approach to the text categorization. The graph-based version of KNN algorithm was empirically validated as its better performance than the traditional version in the text classification. The differences of this research from the previous work are to modify the KNN variants into the graph-based versions and to apply them to the text categorization.

We propose the three graph based KNN variants as the approaches to the text categorization, in this research. The first graph based KNN variant discriminates the nearest neighbors by their distances in voting their labels. The second graph based KNN variant discriminates vertices or edges by their correlations with categories in computing the similarities of a novice text with the sample texts. The last graph based KNN variant discriminates the training examples by their qualities in both voting the labels of nearest neighbors and computing their similarities with a novice example. This research is intended to apply the three graph based KNN variants to the text categorization.

Let us mention some benefits from this research. Converting texts into graphs is the solution the main problems in encoding texts into numerical vectors, such as the huge dimensionality and the sparse distribution. The classification performance is improved by adopting the KNN variants as the better approaches than the KNN algorithm. It is further improved by modifying the KNN variants into the graph-based versions. The graph based KNN variants which process graphs directly and are proposed as the approaches to the text categorization are more recommendable for implementing the text categorization system.

Let us mention the organization of this paper. In Section II, we explore the previous works which are relevant to this study. In Section III, we describe in detail what is proposed in this research. In Section IV, we mention the significances of this research and the remaining tasks. This paper is composed of the four sections.

II. PREVIOUS WORKS

This section is concerned with the previous works which are relevant to this research. It is intended to expand the style of modifying the KNN algorithm to its three variants. The directions of exploring previous works are derivation of KNN variants, modification of the standard KNN algorithm into its graph-based version as an approach to the text classification, and the previous case of representing a text into a graph. The distinguishable points of this research are derivation of three KNN variants from the standard version, modification of them into their graph-based versions, and application of them to the text classification. This article

is intended to explore previous works for providing the background for this research.

Let us explore the previous works on KNN variants. In 2001, the KNN variant where nearest neighbors are discriminated by their distances from or similarities as a novice item was applied to the text classification by Han et al. [1]. In 2008, MKNN (Modified K Nearest Neighbor) the KNN variants where training examples are discriminated by their validness, was proposed by Parvin et al. [4]. In 2018, the KNN variant where the attributes are discriminated by their correlations were proposed by Nababan and Sitompul [8]. However, this research uses different specific metrics for discriminating attributes, nearest neighbors, and training examples.

Let us explore the previous works on applying the modified version of KNN algorithm to the text classification. In 2018, it was initially proposed that the graph-based version of KNN algorithm should be applied to the text classification by Jo [7]. In 2019, the modified KNN algorithm was validated in the task by Jo [9]. The journal article which describes in detail the modified KNN algorithm and its application to the text classification is finally prepared for its publication by Jo [11]. This research is based on the three previous works.

Let us explore the previous works on representation of a text into a graph. In 2004, the process of representing a text into a graph was mentioned by Hensman [2]. In 2007, encoding a text into a graph for doing data mining tasks was covered by Jin and Srihari [3]. In 2020, the matching criteria of graphs which represent texts was proposed by Osman and Barukub [10]. The scheme of encoding a text into a graph and the similarity matrix between graphs may be different in this research.

Let us mention the differences of this research from the previous works which were explored above. The KNN variants in the previous works are numerical vector-based versions, and they will be modified into graph-based versions in this research. As the expansion of [11], we modify and adopt the KNN variants for implementing the text classification system. In this research, we connect representation of a text into a graph to the modification of the KNN variants. The modified KNN variants will be described in detail in Section III.

III. PROPOSED WORKS

This section is concerned with what is proposed in this research. In Section III-A, we describe the process of encoding a text into a graph and the proposed similarity metric. In Section III-B, we describe the standard version of KNN algorithm where the proposed similarity metric is adopted. In Section III-C, we derive the three variants from the standard version. In Section III-D, we present the system architecture of the text classification system.

A. Encoding Process

This section is concerned with the graph as a text representation and the similarity between graphs. In representing a text into a graph, a word is given as a vertex, and a similarity between words is given as an edge. A graph is viewed as a set of edges, and the similarity between edges is computed before computing the similarity between graphs. The similarity between them is computed by averaging the similarities of all possible edge pairs. This section is intended to describe the process of encoding a text into a graph and the process of computing the similarity between graphs.

The process of encoding a text into a graph is illustrated in Figure 1. Words are retrieved as the vertices from the text by indexing it. The similarities among words are computed as edges in the edge definition. In the edge selection, the edges with their higher similarities are selected for representing a text into a graph. Refer to [5] for studying the encoding process in more detail.

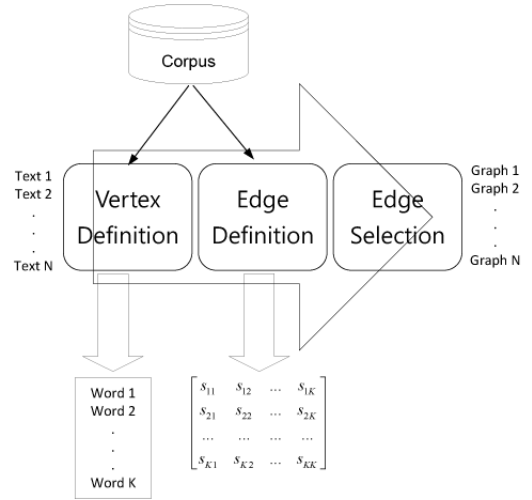


Figure 1. Process of Encoding Texts into Graphs

The three cases of computing the similarity between two edges are illustrated in Figure 2. Each weight which is associated with its own edge is assumed to be a normalized value between zero and one, and if both nodes are identical to each other, the average over two weights is the similarity between two edges. If either of the two nodes is identical to each other, the product of two weights is the similarity between two edges. If neither of two nodes is identical, zero is the similarity between them. The similarity between two edges is always a normalized value between zero and one.

Let us mention the process of computing the similarity between graphs as a semantic similarity between texts. The two graphs which represent texts are notated by edge sets, $G_1 = \{e_{11}, e_{12}, \dots, e_{1|G_1|}\}$ and $G_2 = \{e_{21}, e_{22}, \dots, e_{2|G_2|}\}$. Two edges, $e_{1i} \in G_1$ and $e_{2j} \in G_2$, are associated with their

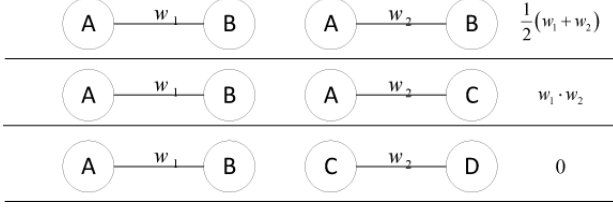


Figure 2. Three Cases of Comparing Two Edges with Each Other

own weights, w_{1i} and w_{2j} , and the similarity between two edges, e_{1i} and e_{2j} by equation (1), (2), or (3), depending on the cases which are mentioned above,

$$\text{sim}(e_{1i}, e_{2j}) = \frac{1}{2}(w_{1i} + w_{2j}) \quad (1)$$

$$\text{sim}(e_{1i}, e_{2j}) = w_{1i} \times w_{2j} \quad (2)$$

$$\text{sim}(e_{1i}, e_{2j}) = 0 \quad (3)$$

The similarity between two graphs is computed by equation (4),

$$\text{sim}(G_1, G_2) = \frac{1}{|G_1| \times |G_2|} \sum_{i=1}^{|G_1|} \sum_{j=1}^{|G_2|} \text{sim}(e_{1i}, e_{2j}). \quad (4)$$

The value which is generated from equation (4), is always given as a normalized value between zero and one as shown in equation (5),

$$0 \leq \text{sim}(G_1, G_2) \leq 1. \quad (5)$$

Let us make some remarks on the encoding process and the computation of the similarity between texts. A text is encoded into a graph by the vertex definition, the edge definition, and the edge weighting. The three cases are considered for computing the similarity between edges. The similarity between graphs is computed by equation (4). In the future research, we consider representing a text into multiple graphs.

B. Initial Version of Graph based KNN Algorithm

This section is concerned with the initial version of graph based KNN algorithm. It was initially proposed as the approach to the word classification by Jo in 2018 [6]. The similarity metric between graphs which was described in the previous section is used for computing the similarity between a novice graph and a training example. The labels of its nearest neighbors are voted with their identical weights for deciding the label of a novice input. This section is intended to describe the initial version of KNN algorithm from which its variants are derived.

The process of computing the similarity between a novice item and a training example is illustrated in Figure 3. The labeled words which are gathered as samples are encoded into graphs, and they are notated by a set, $Tr = \{G_1, G_2, \dots, G_N\}$. A novice word is encoded into a graph

notated by G . Its similarities with the graphs which represent the sample words are computed by equation (4), as $\text{sim}(G, G_1), \text{sim}(G, G_2), \dots, \text{sim}(G, G_N)$. Some training examples which are most similar as the novice input are selected as its nearest neighbors.

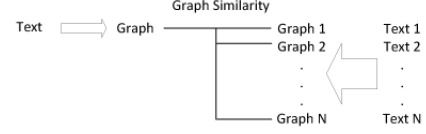


Figure 3. Similarities of Novice Input with Training Examples

In Figure 4, the process of selecting nearest neighbors by the ranking selection is illustrated. The training examples are sorted by the descending order of their similarities, after computing their similarities with a novice example. The training examples with their higher similarities with a novice example are selected as its nearest neighbors, as expressed in equation (6),

$$Nr = \text{Select}_k(Tr, G), Nr \subseteq Tr \quad (6)$$

The set of nearest neighbors, Nr , is composed with elements as expressed in equation (7),

$$Nr = \{G_1^{near}, G_2^{near}, \dots, G_k^{near}\} \quad (7)$$

The elements in the set, Nr , are used for voting their labels in the next step.

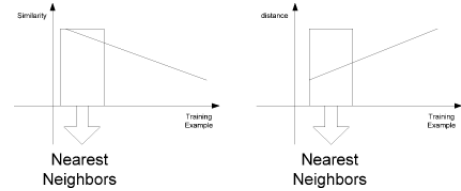


Figure 4. Selection of Most Similar or Least Distant Training Examples as Nearest Neighbors

The process of voting the labels of nearest neighbors for decoding the label of a novice word is illustrated in Figure 5. The predefined categories are notated by $c_1, c_2, \dots, c_m$, and the label of a nearest neighbor is notated by $c_j = \text{label}(G_i^{near})$. The number of nearest neighbors which belong to the category, c_j , is notated by $\text{Count}(Nr, c_j)$. The label of the novice item, G , is decided by equation (8),

$$\text{label}(G) = \arg\max_{j=1}^m \text{Count}(Nr, c_j) \quad (8)$$

The process of deciding the label of a novice item by equation (8), is called voting.

Let us make some remarks on the initial version of KNN algorithm from which we derive some variants. It computes the similarities of a novice item with the training examples. The training examples with their highest similarities with

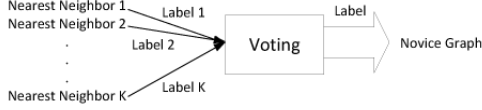


Figure 5. Voting of Labels of Nearest Neighbors for Deciding Label of Novice Input

the novice item are selected as its nearest neighbors. The label of the novice item is decided by voting the labels of its nearest neighbors. The ranking selection is adopted for selecting the nearest neighbors from training examples, in this version.

C. KNN Variants

This section is concerned with the variants which are derived from the KNN algorithm which was covered in Section III-B. The difference from the traditional version is to adopt similarity metric between graphs for computing the similarity between a novice item and a training example. In this section, we derive the three variants by discriminating nearest neighbors, edges, or training examples. The three variants of KNN algorithm are adopted for implementing the text classification system. This section is intended to describe the three variants.

Let us mention the KNN variant which is derived by discriminating the nearest neighbors. A set of nearest neighbors is notated by $Nr = \{G_1^{near}, G_2^{near}, \dots, G_k^{near}\}$, and its elements are assumed as $sim(G, G_1^{near}) \geq sim(G, G_2^{near}) \geq \dots \geq sim(G, G_k^{near})$. The weights, $w_1, w_2, \dots, w_k$, are assigned to the nearest neighbors, $G_1^{near}, G_2^{near}, \dots, G_k^{near}$, following, $w_1 \geq w_2 \geq \dots \geq w_k$, and the total weights are computed for the category, c_j by equation (9),

$$CS(Nr, c_j) = \sum_{G_i \in Nr} w_i \quad (9)$$

The label of the novice item, G , is decided by equation (10),

$$label(G) = \arg\max_{j=1}^m CS(Nr, c_j) \quad (10)$$

There are two ways of assigning weights to the nearest neighbors: exponentially decreasing way and arithmetic decreasing way as shown in Figure 6.

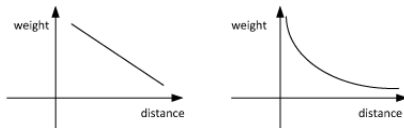


Figure 6. Discrimination over Nearest Neighbors

Let us mention the second graph based KNN variant where the edges are discriminated for computing the similarity between a novice item and a training example as shown in Figure 7. The frequency of the edge, e_i , in the words which

are labeled with the category, c_j , is $f(e_i|c_j)$, and the total frequency of the edge, e_i , in the sample words, is $f(e_i)$. The influence of the edge, e_i , on the category, c_j , is computed by equation (11),

$$I(e_i, c_j) = \frac{f(e_i, c_j)}{f(e_i)}, \quad (11)$$

and the weight, w_i , is computed by equation (12),

$$w_i = \max_{k=1}^m I(e_i, c_j). \quad (12)$$

The equation (4) for computing the similarity between graphs is modified into equation (13),

$$sim(G_1, G_2) = \frac{1}{|G_1| \times |G_2|} \sum_{i=1}^{|G_1|} \sum_{j=1}^{|G_2|} w_i w_j sim(e_{1i}, e_{2j}). \quad (13)$$

In this variant, equation (4) is replaced by equation (13) for computing the similarity between a novice item and a training example.

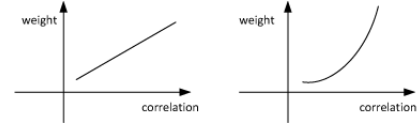


Figure 7. Discrimination over Edges

Let us mention the third KNN variant where the training examples are discriminated for computing the similarity between a novice item and a training example and/or voting the nearest neighbors for deciding the label as shown in Figure 8. The similarity threshold is used as the parameter, and for each training example, other training examples are taken within the threshold from it as its nearest neighbors. The number of nearest neighbors and the number of training examples which are labeled identically to the training example, G_i , are notated respectively by r_i and r_i^- , and the weight is computed by equation (14),

$$w_i = \frac{r_i^-}{r_i} \quad (14)$$

The similarity between the novice item, G , and the training example, G_i is computed by equation (15),

$$sim(G, G_i) = \frac{w_i}{|G| \times |G_i|} \sum_{j=1}^{|G|} \sum_{k=1}^{|G_i|} sim(e_j, e_{ik}). \quad (15)$$

and the total weight is computed for the category, c_j , by equation (16), in voting,

$$CS(Nr, c_j) = \sum_{G_i \in Nr} w_i. \quad (16)$$

Equation (15) and (16) are used respectively for computing the similarity between a novice item and a training example and voting the nearest neighbors for each category.

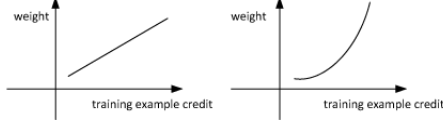


Figure 8. Discrimination over Training Examples

Let us make some remarks on the three variants of KNN algorithm which are proposed as the approaches to the text classification. In the first variant, the nearest neighbors are discriminated for voting their labels. In the second variant, the edges are discriminated for computing the similarity between a novice input and a training example. In the third variant, the training examples are discriminated for voting the labels of the nearest neighbors and/or computing the similarity between graphs. We may consider the trainable KNN algorithm which optimizes the weights of the nearest neighbors, the edges, and the training examples for minimizing the training error.

D. System Architecture

This section is concerned with the architecture and the execution flow of the text classification system. In Section III-C, we described the graph based KNN variants which are adopted for implementing the text classification system. In the architecture of text classification system, which was previously proposed, the initial KNN algorithm which is mentioned in Section III-B is replaced by its variants. The process of executing the program is to encode a text into a graph and classify it into one among the predefined categories. This section is intended to describe the text classification system which is implemented in this study.

The collection of sample texts for building the training set is illustrated in Figure 9. The topics are predefined a list of topic 1, topic 2, ..., topic M; in implementing the system, it is assumed that the text classification belongs to the flat classification where the predefined categories are given as a list. For each topic, texts about it are collected and encoded into graphs by the process which is described in Section III-A. The M groups of numerical vectors which are shown in the bottom of Figure 9 are given as the training set. The hierarchical text categorization is considered in implementing the next version of the text classification system.

The architecture of the text classification system which consists of the encoding module, the similarity computation module, and the voting module is illustrated in Figure 10. The encoding module encodes texts into graphs by the process which is described in Section III-A. The similarity computation module computes the similarities of a novice text with the sample texts and selects ones with their highest similarities as the nearest neighbors as the core part in the system. The voting module decides the label of the novice item by voting the labels of its nearest neighbors.

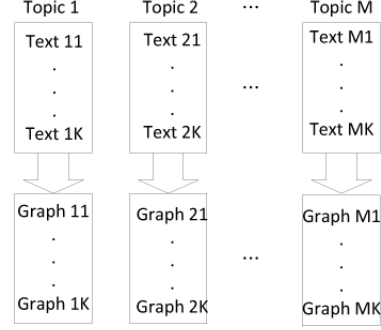


Figure 9. Preparation of Training Examples

The difference from the architecture which was proposed in [9] is to replace the initial version of the KNN algorithm by its variants.

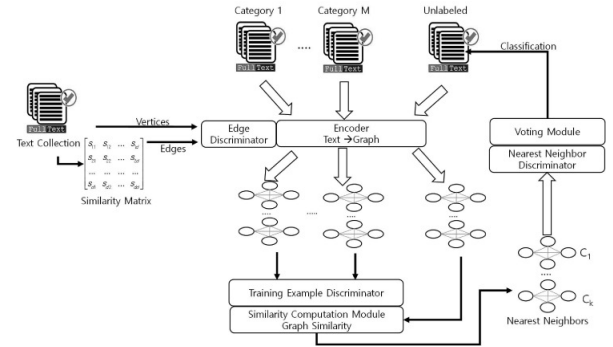


Figure 10. System Architecture

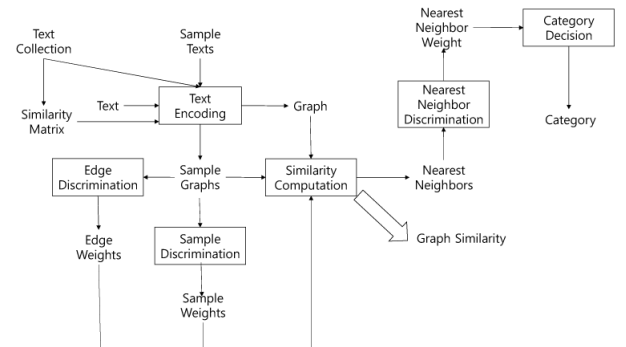


Figure 11. System Flow

Let us make some remarks on the proposed system which illustrated in Figure 10 as its architecture. The M topics are predefined, texts, each of which is labeled with one among the topics, are collected, and they are encoded into graphs. Novice texts are classified by one of the three KNN variants which are described in Section III-C. The difference from the previous version of the proposed system is the ability to select the nearest neighbor discrimination, the edge

discrimination, or the training example discrimination. The better performance of the text classification is expected by replacing the initial version by its variants.

IV. CONCLUSION

Let us mention the significances of this research as the conclusion. We apply the proposed similarity metric to the KNN variants as well as the standard version. We derive the three variants from the standard version by discriminating the nearest neighbors, the attributes, and the training examples. We design the text classification system by adopt the KNN variants with the proposed similarity metric. In the next research, we will implement the text classification system as a real system.

Let us mention the remaining tasks as the further discussions on this research. We need to improve the speed of computing the similarity between graphs in the implementation level. We modify other machine learning algorithms such as the Naïve Bayes and the SVM (Support Vector Machine) into the graph-based versions. We may apply the proposed versions of KNN variants to other tasks such as the text segmentation and the text summarization. The text categorization may be implemented by adopting the proposed approaches as real programs.

REFERENCES

- [1] E.H. Han, G. Karypis and V. Kumar, "Text Categorization Using Weight Adjusted k-Nearest Neighbour Classification" pp53-64, in *Advances in Knowledge Discovery and Data Mining. PAKDD 2001*, Berlin, Heidelberg:Springer, 2035, 2001.
- [2] S. Hensman, "Construction of conceptual graph representation of texts", *Proceedings of the Student Research Workshop at HLT-NAACL 2004*. 2004.
- [3] W. Jin and R. K. Srihari "Graph-based text representation and knowledge discovery", 807-811, *The Proceedings of ACM symposium on Applied computing* 2007.
- [4] H. Parvin, A. Hosein, and M.B. Behrouz, "MKNN: Modified k-nearest neighbor" *Proceedings of the World Congress on Engineering and Computer Science. Vol. 1. Newswood Limited*, 2008.
- [5] T. Jo, "Graph based KNN for Optimizing Index of News Articles", 53-62, *Journal of Multimedia Information System*, 3, 2016.
- [6] T. Jo, "Comparing Graph based K Nearest Neighbor with Traditional Version in Word Categorization in NewsPage.com, 12-18, *International Journal of Advanced Social Sciences*, 1, 2018.
- [7] T. Jo, "Graph based KNN for Text Categorization", 260-264, *The Proceedings of IEEE 18th International Conference on Advanced Communication Technology*, 2018.
- [8] A.A. Nababan and O. S. Sitompul, "Attribute weighting based K-nearest neighbor using Gain Ratio", *Journal of Physics: Conference Series*, 1007, 1, 2018.
- [9] T. Jo, "Graph based Version of K Nearest Neighbor for classifying News Articles", 4-7, *The Proceedings of International Conference on Green and Human Information Technology Part I*, 2019.
- [10] A.H. Osman and O.M. Barukub, "Graph-based text representation and matching: A review of the state of the art and future challenges", 87562-87583, *IEEE Access*, Vol 8, 2020.
- [11] T. Jo, "Graph Similarity Metric for Modifying K Nearest Neighbor for Classifying Texts", in Preparation, 2023.